FAST AND EFFECTIVE TEXT DETECTION

Xiaojun Li[1], *Weiqiang Wang*[1,2], *Shuqiang Jiang*[2], *Qingming Huang*[1,2], *Wen Gao*[2,3]

[1]Graduate University of Chinese Academy of Sciences, Beijing, China

[2] Key Lab of Intell. Info. Process., Inst. of Comput. Tech., Chinese Academy of Sciences, Beijing China

[3]Institute of Digital Media, Peking University, Beijing, China

{xjli, wqwang, sqjiang, qmhuang, wgao}@jdl.ac.cn

ABSTRACT

Text in images and videos is a significant cue for visual content understanding and retrieval. In this paper, we present a fast and effective approach to locate text lines even under complex background. First, our algorithm uses the stroke filter to calculate the stroke maps in horizontal, vertical, left-diagonal, right-diagonal directions. Then a 24-dimensional feature is extracted for each sliding window and a SVM is used to obtain rough text regions. The rough text regions are further refined through a group of rules. And candidate text lines were localized more accurately through projection profile of the refined text regions. Finally another SVM classifier based on a 6-dimensional feature is used to verify the candidate text lines. The experimental results on challenging databases show that this approach can fast and effectively detect and localize text lines.

Index Terms— Text Detection, Stroke Filter, SVM

1. INTRODUCTION

Text embedded in images and videos provides brief and important information for visual information summarization and indexing. In recent years a lot of research works focused on detecting and localizing text in images or videos. Many characteristics of text regions have been summarized and characterized by effective features. For instance, embedded text usually goes with dense edges [1]; text pixels almost have homogeneous color [2]; character strokes form distinct textures [3][4]. Liu *et al.* [5] analyzed the properties of character strokes and proposed to use stroke filter to directly detect and localize text. Based on different features, there are two main categories of modeling methods. One is constraint-based methods, such as [1][5][6][7] *et al.*; the other is learning-based methods, such as neural network [8][9], SVM [3][4][5][10]. The methods in the first category require designers themselves to finish the task of summarizing the effective classification rules, so it is not easy for engineers to create text detector with good performance. Nowadays the learning-based methods become very popular.

In this paper, Inspired by [5], we utilize the stroke filter to obtain the low-level representation of image content, and then derive more compact classification feature to create a fast and effective text detector in a machine learning way. Compared with [4], our algorithm investigates to use more simple and distinctive features, so it has higher computation efficiency. Different from Liu *et al.* [5] which used the constraint-based way (connected component analysis) to detect text, our algorithm employs the SVM to identify text regions.

The remaining parts of this paper are organized as follows. Section 2 presents our text detection approach in detail. In section 3, we report our comparison experiment results on two challenging datasets. Section 4 concludes the whole paper.

2. OUR ALGORITHM

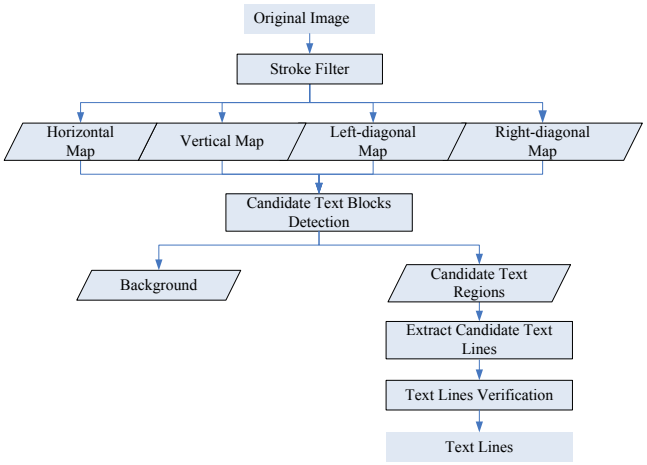


Fig.1. Flow chart of the proposed algorithm

Fig.1 summarizes the flow chart of the proposed algorithm.

For an original image, we first use the stroke filter [5] to obtain four stroke maps which characterize the stroke strengths in horizontal, vertical, left-diagonal, right-diagonal

directions. Then the corresponding features are extracted for each sliding window, and a SVM is employed to classify the sliding windows into text blocks and non-text blocks. All the text blocks form the candidate text regions represented by a binary mask image. Further some rules are designed to extract candidate text lines through filling and merging operation, and removing obvious non-text rectangles. According to the candidate text lines, finally a new feature is extracted from each candidate text line, and a second SVM is employed to decide the class label of the candidate text line, i.e., text or non-text.

2.1. Generating Stroke Maps

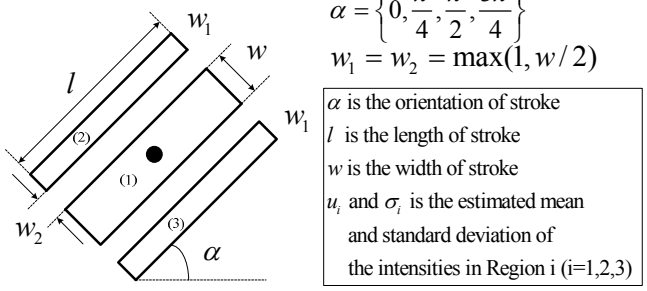


Fig.2. An illustration of a stroke filter

For a color image, it is first converted into a grayscale image. Then, a group of stroke filters are applied to generate stroke maps corresponding to the horizontal, vertical, left-diagonal, right-diagonal directions. As shown in Fig.2, the response $R_{\alpha,l,w}(x,y)$ of a stroke filter at the central point (x,y) depends on the pixel values in three rectangular regions, where α, l and w are three parameters which correspond to the orientation, length and width of a stroke respectively. Concretely, the stroke filter response is defined as:

$$R_{\alpha,l,w}(x,y) = \frac{|u_1 - u_2| + |u_1 - u_3| - |u_2 - u_3|}{\max(\sigma_1, 2)} . \quad (1)$$

More information about the stroke filter can be found in [5].

In our method, the stroke filter responses corresponding to four different directions are calculated respectively by:

$$R_\alpha(x,y) = \max_{l,w}\{R_{\alpha,l,w}(x,y)\} . \quad (2)$$

Through formula (2), we obtain the stroke maps for the horizontal ($\alpha = 0$), vertical ($\alpha = \pi/2$), left-diagonal ($\alpha = 3\pi/4$), right-diagonal ($\alpha = \pi/4$) directions.

2.2. Candidate Text Blocks Detection

Through a $W * H$ sliding window, a SVM is used to identify whether a potential text block exists at the position covered by the sliding window B . Correspondingly, a 24-dimensional feature is extracted from the four stroke maps for each sliding window. Since text blocks usually result in significant response values in four stroke maps and non-text blocks have not significant response values in all maps. We employ the statistical features in stroke maps to capture these properties. Concretely, the features include the mean m_α, the variance v_α, and the weighted energy e_α :

$$m_\alpha = \frac{1}{W * H} \sum_{(x,y)\in B} R_\alpha(x,y) , \quad (3)$$

$$v_\alpha = \frac{1}{W * H} \sum_{(x,y)\in B} (R_\alpha(x,y) - m_\alpha)^2 , \quad (4)$$

$$e_\alpha = \frac{1}{W*H} \sum_{(x,y)\in B} \frac{1}{1+(x-x_c)^2+(y-y_c)^2} R_\alpha^2(x,y) , \quad (5)$$

where $R_\alpha(x,y)$, $\alpha \in \{0, \pi/4, \pi/2, 3\pi/4\}$ denotes the response values at the stroke maps for four different directions, B denotes the region covered by the sliding window, and (x_c, y_c) is the coordinates of the central location of B . Formula (5) shows those intensities more close to the central point of the sliding window have greater weights.

To characterize the spatial distribution of strokes, we define the corresponding features, vertical accumulation profile (VAP) and horizontal accumulation profile (HAP). For each sliding window in the vertical stroke map, it is vertically equally partitioned into eight rectangle regions. In each rectangle $S_i, i = 1,2,...,7$, the VAP is calculated as follows:

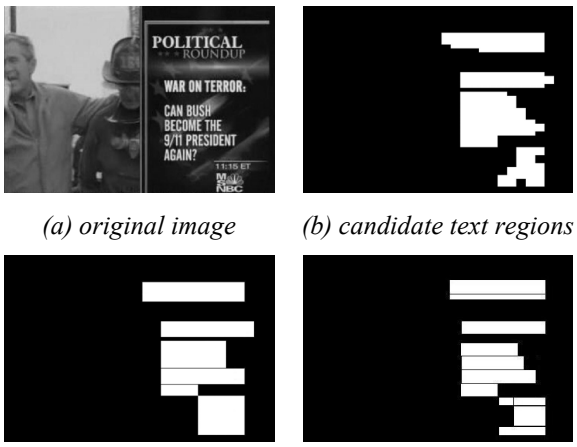
$$VAP[i] = \sum_{(x,y)\in S_i} R_{\frac{\pi}{2}}(x,y) . \quad (6)$$

Similarly, for each sliding window in the horizontal stroke map, it is horizontally equally partitioned into four rectangle regions. In each rectangle $M_j, j = 1,2,3,4$, the HAP is calculated as follows:

$$HAP[j] = \sum_{(x,y)\in M_j} R_0(x,y) . \quad (7)$$

So each block covered by the sliding window is represented by a 24-dimensional feature vector.

Compared with other classifiers, such as the neural network and the decision tree etc., the SVM needs fewer training samples and has better generalization ability, so we select the SVM as our classifier to get candidate text blocks. The SVM in the stage was trained on a dataset consisting of 240 text blocks and 480 non-text blocks in our work. If the output of the SVM classifier is positive, the pixels in the sliding window will be totally labeled as text pixels. The moving step of the sliding window is horizontally $W/2$ and vertically $H/2$. As a result, we create a binary mask image whose white regions represent the candidate text regions and black region represents the background. Fig. 3(b) gives an example of the output result after the stage.



(a) original image *(b) candidate text regions*

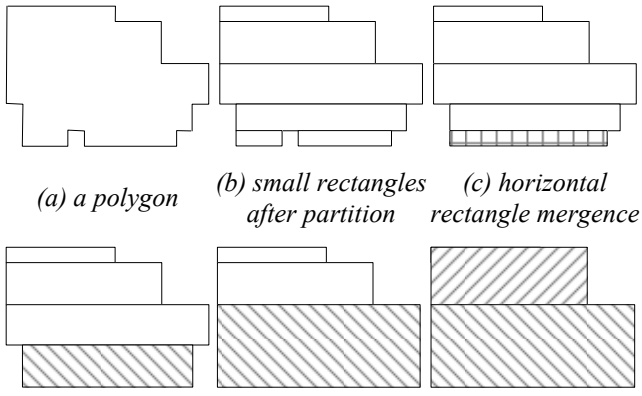
(c) regular rectangles *(d) candidate text lines*

Fig.3 Rough text detection

2.3. Extract Candidate Text Lines

As shown in Fig. 3(b), the candidate text regions may cover a few non-text regions, and multiple boundary rectangles corresponding to text regions are often connected together. Thus, we first use the following computation steps to partition those connected polygons into regular rectangles (Fig. 3(c)):

1) Partition a polygon into small rectangles (Fig. 4(b)).
2) If the gap between two rectangles in a horizontal line is less than 1/6 of the total width of them, fill the gap to merge them together (Fig. 4(c)).
3) For two vertically adjacent rectangles, if the width of a shorter rectangle exceeds 4/5 of the longer one, merge them into a larger rectangle whose height is the sum of their heights and width is the maximum of them (Fig. 4(d,e)).
4) If the height of a rectangle is less than 1/3 of its vertically adjacent rectangle, merge them in the same way as the step (3) does (Fig. 4(d)).



(a) a polygon *(b) small rectangles after partition* *(c) horizontal rectangle mergence*

(d) first vertical rectangle mergence *(e) second vertical rectangle mergence* *(f) Third vertical rectangle mergence*

Fig.4 Partition the connected polygon

Then for each regular rectangle generated, we use the horizontal and vertical projection method to efficiently localize text lines. Our method is similar to that in [1], but only one pass of projection is involved. The intensity values used in projection evaluation are the sum of responses in four stroke maps. For generated boundary rectangles corresponding to text lines, some obvious non-text rectangles are removed, if their heights do not belong to a reasonable range $[\alpha,\beta]$, or their aspect ratio exceeds a certain threshold γ. Fig. 3(d) shows the final candidate text lines after projection operation.

2.4. Text Lines Verification

For each candidate text line C, a 6-dimensional feature is extracted and used by a new SVM classifier to finally verify whether it is a true text line. The feature includes the mean m, standard deviation v, and four features which reflect statistical distribution of horizontal stroke accumulation strengths in the text line (more analyses can be found in [4][8]), i.e.,

$$
\begin{aligned}
m &= \frac{1}{W * H} \sum_{(x,y)\in C} \sum_{\alpha} R_{\alpha}(x,y), \\
v &= \frac{1}{W * H} \sum_{(x,y)\in C} \Big(\sum_{\alpha} R_{\alpha}(x,y) - m\Big)^2, \\
c(\rho) &= \frac{1}{W} \sum_{x=1,\ldots,W} (V(x) > \rho * u), \\
&\rho \in \{1.4, 1.0, 0.6, 0.2\}, \\
&V(x) = \sum_{y=1}^{H} \sum_{\alpha} R_{\alpha}(x,y), \; u = \frac{1}{W} \sum_{x=1,\ldots,W} V(x),
\end{aligned}
\tag{8}
$$

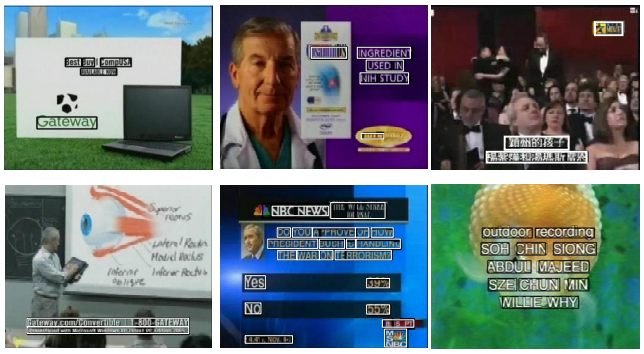
where W and H denote the width and the height of the text region. The SVM in the stage was trained on a dataset of consisting of 200 text lines and 160 non-text lines in our work. Fig. 5(b,c) show the results before and after text lines verification.

(a) original images *(b) candidate text lines* *(c) verified text lines*

Fig.5 Text line verification

3. EXPERIMENTAL RESULTS

Two challenging test sets are selected to evaluate our approach. One consists of 308 images from the Web, recorded broadcast videos, or digital videos. Each image in it is carefully selected so that the embedded text has different sizes, colors, languages, and background textures etc. The other is Microsoft common test set containing 46 images [11]. In our experiments, the length and width of the stroke filter has two groups of data, i.e., (1) $l = 3, w = 1$ (2) $l = 4, w = 3$, the size of the sliding window was set as $W = 24$, $H = 12$, and the parameters α, β and γ were set as 6, 72 and 1.2 respectively. Fig.6 shows some experimental results.



(a) images *(b) covers* *(c) video frames*

Fig.6 Some experimental results

For quantitative comparisons, three metrics are adopted to evaluate the performance of our method, i.e., (1) ***Speed:*** The average number of processed images per second for text detection and localization; (2) ***Recall:*** The ratio of the text regions correctly detected to the ground-truth text regions; (3) ***Accuracy****:* The ratio of the text regions correctly detected to the text regions claimed by the system. A correct detection is counted if and only if the intersection of a detected text region (DTR) and a ground-truth text region (GTR) covers at least 90% of both the DTR and the GTR. The GTRs in the test images are localized manually.

Table 1. Performance evaluation of three approaches

Approach	Speed(images/s)	Recall	Accuracy
Our approach	12.9	91.1%	95.8%
Ye *et al.* [4]	10.1	90.8%	90.3%
Liu *et al.* [5]	11.7	91.3%	92.4%

The related experimental results are summarized in Table I. Compared with the method presented in [4] and [5], our method has a better performance in detection speed and accuracy. The reason for fast speed is that region growing or connected component analysis cost much time. The reason for high accuracy is that the learning-based method is easier to obtain better classification models than constraint-based method in distinguishing text and non-text regions and two sequential simple classifiers achieve better performance than one complex classifier.

4. CONCLUSION

In this paper, a fast and effective text detection approach is proposed. The devised features based on the stroke maps in four directions are able to better represent the intrinsic characteristic of text. The machine learning based method can construct text detector of high performance more easily. Combination of the stroke-related feature and machine learning modeling methods is a promising solution to the issue of text detection in images and videos.

5. ACKNOWLEDGEMENTS

This work was supported in part by National Hi-Tech Development Program of China under Grant 2006AA01Z117 and 2006AA010105, by National Key Technologies R&D Program under Grant 2006BAH02A24-2 and research start-up fund of GUCAS, and by National Natural Science Foundation of China under Grant 60773136 and 60702035.

6. REFERENCES

[1] M.R Lyu, J.Song and M.Cai, "A comprehensive method for multilingual video text detection, localization, and extraction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 2, pp. 243-255, Feb. 2005.

[2] V. Y. Mariano and R. Kasturi, "Locating uniform-colored text in video frames," *in Proc. 15th Int. Conf. Pattern Recognit.*, vol. 4, pp. 539–542, Barcelona, Spain, Sep. 2000.

[3] C. Zhu, W. Wang and Q. Ning, "Text detection in images using texture feature from strokes", *7th Pacific-Rim Conf. on Multimedia*, Hangzhou, China, Nov. 2006, pp. 295-301.

[4] Q.Ye, Q.Huang, W.Gao and D. Zhao, "Fast and robust text detection in images and video frames," *Image Vis. Comput.* vol. 23, no. 6, pp. 565-576, Jun. 2005.

[5] Q. Liu, C. Jung, S. Kim, Y. Moon and J.Kim," Stroke filter for text localization in video images," in *Proc. Int. Conf. Image Process.,* Atalanta, GA, USA, Oct. 2006, pp. 1473-1476.

[6] N. Otsu, "A threshold selection method from gray-scale histograms," *IEEE Trans. Syst., Man, Cygbernet.,* vol. SMC-9, no. 1, pp. 62-66, Jan. 1979.

[7] C. Wolf, and JM. Jolion, "Extraction and recognition of artificial text in multimedia documents," *Pattern Anal. Applicat.*, vol. 6, no. 4, pp. 309-326, Feb. 2004.

[8] R. Lienhart and A. Wernicke, "Localizing and segmenting text in images and videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol.12, no.4, pp. 256-268, Apr. 2002.

[9] H. Li, D. Doermann, and O. Kia, "Automatic text detection and tracking in digital video," *IEEE Trans. Image Process.*, vol. 9, no. 1, pp. 147-156, Jan. 2000.

[10] D. T. Chen, H. Bourlard, and J-P. Thiran, "Text identification in complex background using SVM," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* Kauai, Hawaii, Dec. 2001, pp. 621-626.

[11] X.S. Hua, W.Y. Liu, and H.J. Zhang, "An automatic performance evaluation protocol for video text detection algorithms," *IEEE Trans. Circuits Syst. Video Technol.* vol. 14, no. 4, pp. 498–507, Apr. 2004.